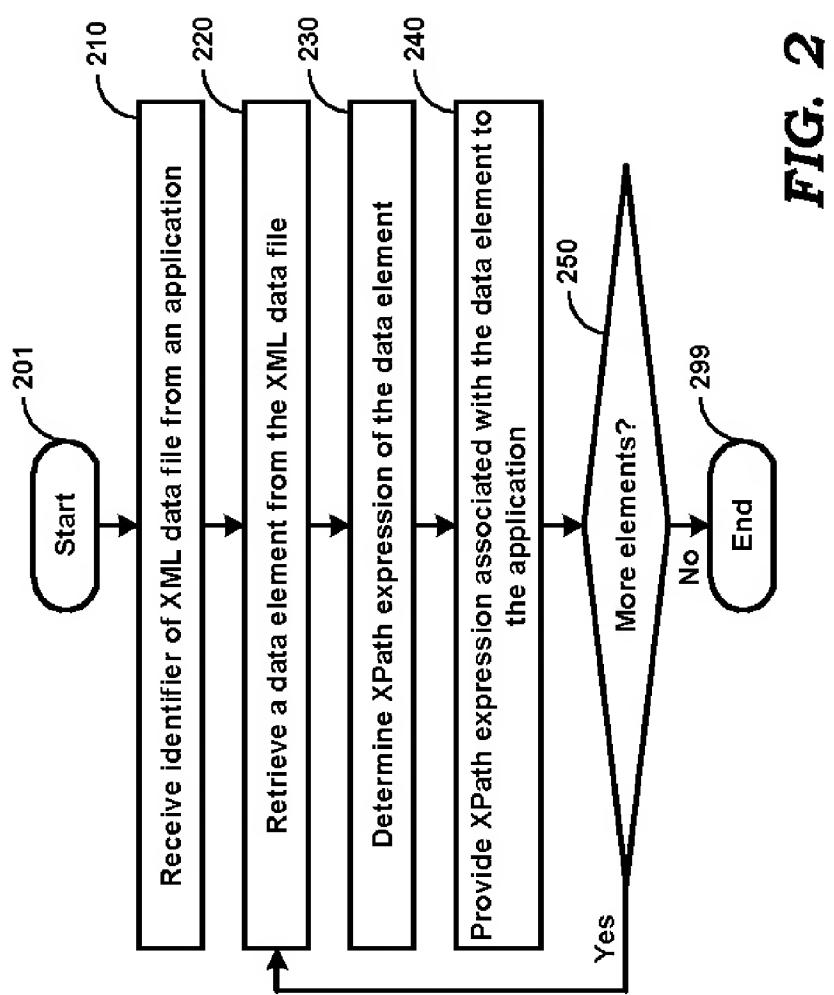
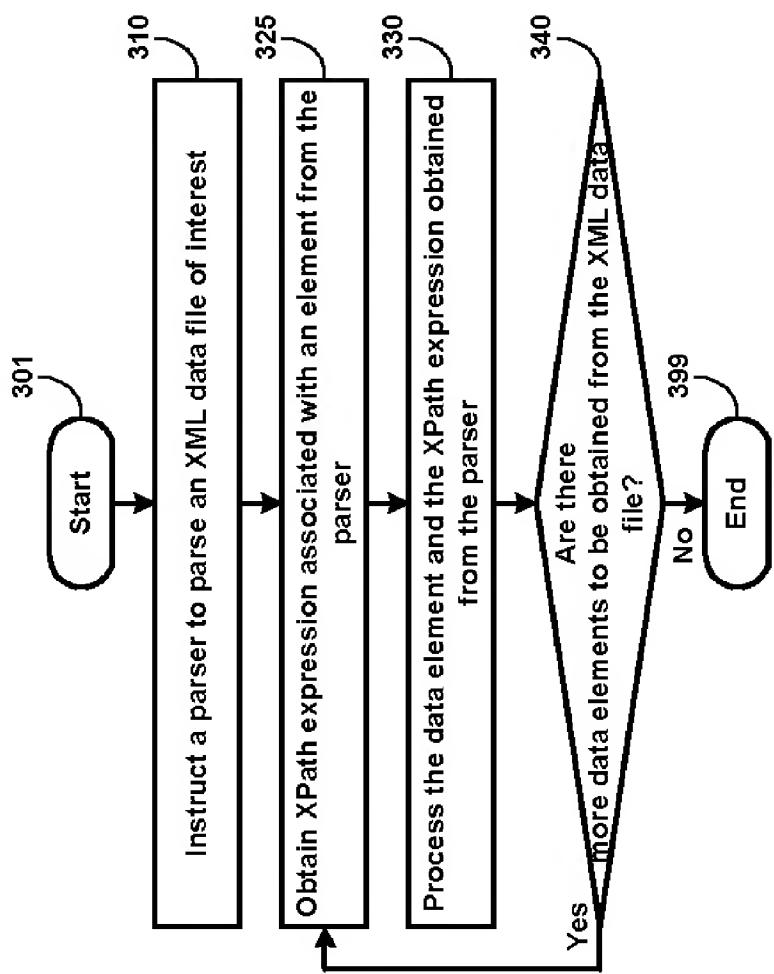


**FIG. 1**



**FIG. 2**



**FIG. 3**

```
401: <?xml version="1.0" encoding="US-ASCII"?>
402: <Books>
403:   <Book isbn="123">
404:     <Name>Sherlock Holmes-I</Name>
405:     <Author>Arthur Conan Doyle</Author>
406:   </Book>
407:   <Book isbn="456">
408:     <Name>Sherlock Holmes-II</Name>
409:     <Author>Arthur Conan Doyle</Author>
410:   </Book>
411:   <Book isbn="789">
412:     <Name>Sherlock Holmes-III</Name>
413:     <Author>Arthur Conan Doyle</Author>
414:   </Book>
415: </Books>
```

**FIG. 4A**

Node (420)	XPath (440)
<b>Books</b>	<i>/Books</i>
<b>Book</b>	<i>/Books/Book[1]</i>
<b>isbn="123"</b>	<i>/Books/Book[1]/@isbn</i>
<b>Name</b>	<i>/Books/Book[1]/Name</i>
<b>Author</b>	<i>/Books/Book[1]/Author</i>
<b>Book</b>	<i>/Books/Book[2]</i>
<b>isbn="456"</b>	<i>/Books/Book[2]/@isbn</i>
<b>Name</b>	<i>/Books/Book[2]/Name</i>
<b>Author</b>	<i>/Books/Book[2]/Author</i>
<b>Book</b>	<i>/Books/Book[3]</i>
<b>isbn="789"</b>	<i>/Books/Book[3]/@isbn</i>
<b>Name</b>	<i>/Books/Book[3]/Name</i>
<b>Author</b>	<i>/Books/Book[3]/Author</i>

**FIG. 4B**

```
501: import javax.xml.parsers.*;
505: import org.w3c.dom.*;
507: import java.util.Vector;
508: public class DOMParsing {
509:     static Vector xpaths= new Vector();
511:     public static void main(String[] args) throws Exception {
513:         DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
517:         DocumentBuilder db = dbf.newDocumentBuilder();
520:         Document doc = db.parse("file:///D:/\\something.xml");
522:         traverse(doc);
523:         for (int i=0; i<xpaths.size(); i++) {
524:             System.out.println("XPath='"+xpaths.get(i));
525:         }
528:     }
529: }
```

**FIG. 5A**

```
550: private static void traverse(Node node) {  
551:     xpaths.add(node.getXPath());  
552:     if (node.getNodeType() == Node.DOCUMENT_NODE) {  
553:         traverse((Document)node).getDocumentElement();  
554:     }  
555:     if (node.getNodeType() == Node.ELEMENT_NODE) {  
556:         NodeList nodes = node.getChildNodes();  
557:         for(int i = 0; i < nodes.getLength(); i++) {  
558:             traverse(nodes.item(i));  
559:         }  
560:     }  
561:     else if (node.getNodeType() == Node.TEXT_NODE) {  
562:         String str = node.getNodeValue();  
563:     }  
564: }  
565: }  
566: }  
567: }  
568: }  
569: }  
570: }  
571: }  
572: }  
573: }  
574: }  
575: }  
576: }  
577: }  
578: }  
579: }  
580: }  
581: }  
582: }  
583: }  
584: }  
585: }  
586: }  
587: }  
588: }  
589: }  
590: }  
591: }
```

**FIG. 5B**

```

601: import javax.xml.parsers.*;
603: import org.xml.sax.*;
605: import org.xml.sax.helpers.*;
606: import java.util.Vector;
607: public class SAXParsingXPath2 extends DefaultHandler {
609:     public static void main(String[] args) throws Exception {
611:         SAXParserFactory spf = SAXParserFactory.newInstance();
613:         SAXParser sp = spf.newSAXParser();
615:         SAXParsingXPath2 handler = new SAXParsingXPath2();
616:         sp.parse("file:///D:\\\\xpath\\\\something.xml", handler);
617:         for (int i=0; i<xpaths.size(); i++) {
618:             System.out.println("XPath=" + xpaths.get(i));
619:         }
620:     }
621:     public void startElement(String uri, String localName, String qName, Attributes attributes, String xpath) {
623:         System.out.println("Start Element='"+qName);
625:         xpaths.add(xpath);
627:     }
629:     public void characters(char[] ch, int start, int length, String xpath) {
631:         String text = new String(ch, start, length);
633:         xpaths.add(xpath);
635:     }
637:     public void endElement(String uri, String localName, String qName, String xpath) {
639:         System.out.println("End Element='"+qName);
643:     }
645: }

```

**FIG. 6**

```

701: import java.io.*;
702: import javax.xml.stream.*;
703: import javax.xml.stream.events.*;
704: import java.util.Vector;
705: public class PullParsingXPath {
706: static Vector xpaths = new Vector();
707: public static void main(String args[]) throws Exception {
708: FileInputStream xmlFile = new FileInputStream("D:\\Rahul\\xpath\\something.xml");
709: XMLInputFactory xif = XMLInputFactory.newInstance();
710: XMLStreamReader pullParser = xif.createXMLStreamReader(xmlFile);
711: while(pullParser.hasNext()) {
712: int eventType = pullParser.next();
713: xpaths.add(pullParser.get XPath());
714: if (eventType == XMLEvent.START_ELEMENT) {
715: System.out.print("<" + pullParser.getName() + ">");
716: }
717: else if (eventType == XMLEvent.CHARACTERS) {
718: System.out.print(pullParser.getText());
719: }
720: else if (eventType == XMLEvent.END_ELEMENT) {
721: System.out.print("</" + pullParser.getName() + ">");
722: }
723: }
724: }
725: }
726: for (int i=0; i<xpaths.size(); i++) {
727: System.out.println("XPath=" + xpaths.get(i));
728: }
729: }
730: }
731: }
732: }
733: }
734: }
735: }

```

**FIG. 7**

XPathParserFactory	
XPathParserFactory	newInstance()
XPathPushParser	createXPathPushParser()
boolean	hasFeature(String feature)
void	setFeature(String feature, Boolean val)
boolean	getFeature(String feature)
void	setProperty(String property, Object val)
Object	getProperty(String property)

**FIG. 8A**

XPathPushParser	
void	setXpathContentHandler(XpathContentHandler)
Xpath ContentHandler	getXpathContentHandler()
void	setErrorHandler(Errorhandler handler)
ErrorHandler	getErrorHandler()
void	parse(InputSource xmlSource)

**FIG. 8B**

XPathContentHandler	
void	startDocument()
void	emit(String xpath, String xpath Val, int eventType, NamespaceResolver nsResolver)
void	emit(String xpath, String xpath Val, int eventType, XPathAttributes attrs, NamespaceResolver nsResolver)
void	endDocument()

**FIG. 8C**

NamespaceResolver	
String	resolveNamespacePrefix(java.lang.String.prefix) 861

**FIG. 8D**

XPathAttributes	
int	getLength() 871
XPathAttribute	item(int index) 872

**FIG. 8E**

XPathAttribute	
String	getXPath() 891
String	getValue() 892

**FIG. 8F**

```

901: import javax.xml.parsers.*;
902: import org.xml.sax.*;
903: import org.xml.sax.helpers.*;
904: import java.util.Vector;
905: public class PushParsingXPath implements XPathContentHandler
906: static Vector xpaths = new Vector();
907: public static void main(String[] args) throws Exception {
908: XPathParserFactory xpf = XPathParserFactory.newInstance();
909: XPathPushParser xpp = xpf.createXPathPushParser();
910: xpp.setFeature("http://xpath-parser/features/abbreviated-xpath", false);
911: xpp.setFeature("parser.xpath.attributes", true);
912: XPathContentHandler xpcHandler = new MyContentHandler();
913: xpp.setContentHandler(xpcHandler);
914: XPathErrorHandler xpeHandler = new MyErrorHandler();
915: xpp.setErrorHandler(xpeHandler);
917: InputSource is = new InputSource("file:///d:\\xpath\\samples\\abc.xml");
919: xpp.parse(is);
920: }
921: public void emit(String xpathExpr, String xpathVal, int eventType, NamespaceResolver nsResolver) {
923:     xpaths.add(xpathExpr);
925: }
927: public void emit(String xpathExpr, String xpathVal, int eventType, XPathAttributes attrs, NamespaceResolver nsResolver) {
929:     xpaths.add(xpathExpr);
931: }
933: }

```

**FIG. 9**

Node (1010)	XPath (1020)	Value (1030)
Books	/Books	null
Book	/Books/Book[1]	null
isbn="123"	/Books/Book[1]@isbn	123
Name	/Books/Book[1]/Name	Sherlock Holmes- I
Author	/Books/Book[1]/Author	Arthur Conan Doyle
Book	/Books/Book[2]	null
isbn="456"	/Books/Book[2]@isbn	456
Name	/Books/Book[2]/Name	Sherlock Holmes- II
Author	/Books/Book[2]/Author	Arthur Conan Doyle
Book	/Books/Book[3]	null
isbn="789"	/Books/Book[3]@isbn	789
Name	/Books/Book[3]/Name	Sherlock Holmes- III
Author	/Books/Book[3]/Author	Arthur Conan Doyle

**FIG. 10A**

Node (1051)	XPath (1052)	Value (1053)	XPathAttributes (1054)
Books	/Books	null	1061
Book	/Books/Book[1]	null	1062
Name	/Books/Book[1]/Name	Sherlock Holmes - I	1063
Author	/Books/Book[1]/Author	Arthur Conan Doyle	1064
Book	/Books/Book[2]	null	1065
Name	/Books/Book[2]/Name	Sherlock Holmes - II	1066
Author	/Books/Book[2]/Author	Arthur Conan Doyle	1067
Book	/Books/Book[3]	null	1068
Name	/Books/Book[3]/Name	Sherlock Holmes- III	1069
Author	/Books/Book[3]/Author	Arthur Conan Doyle	1070

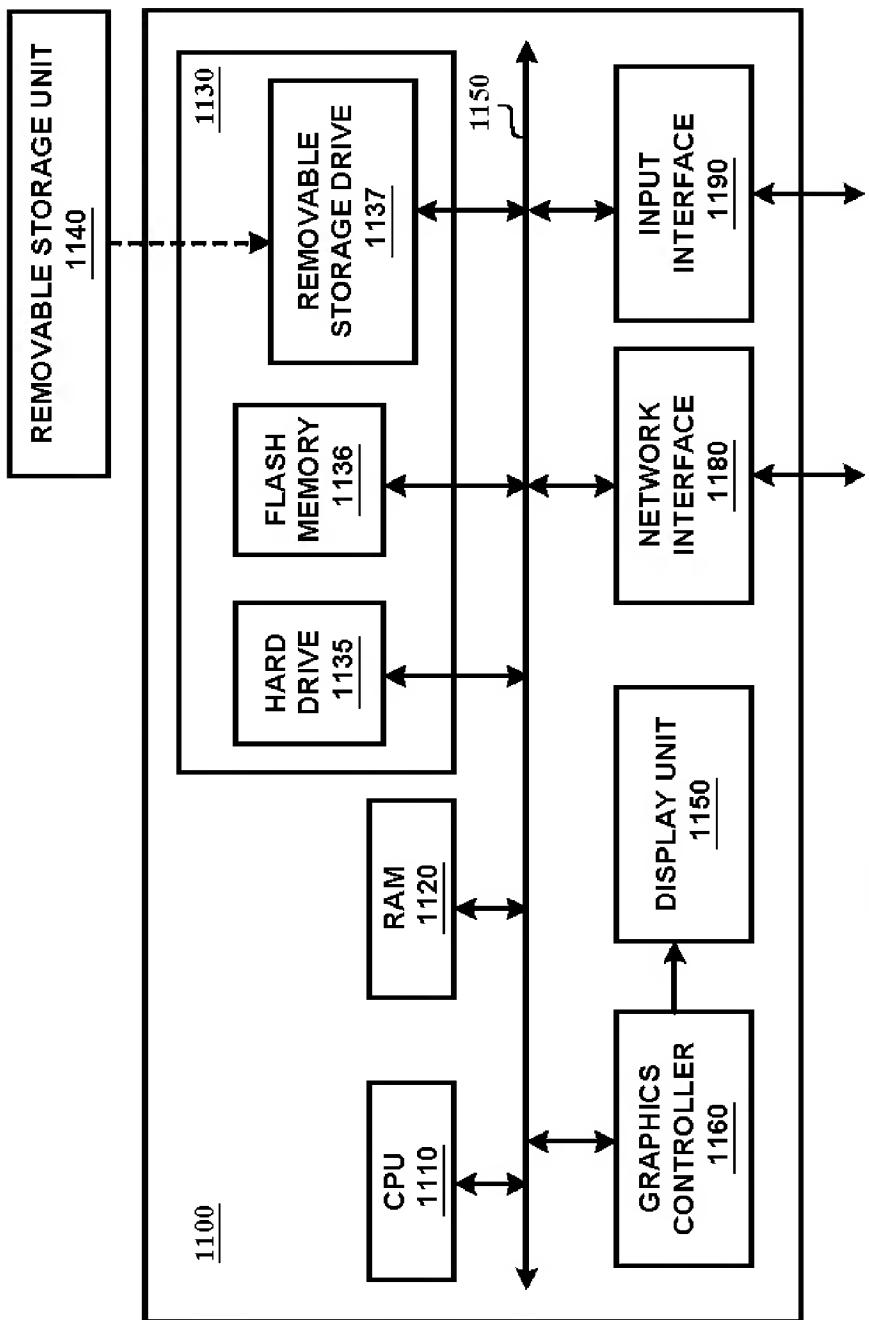
**FIG. 10B**

Node	XPath	Value	XPathAttributes
Books	/child::Books	null	1071
Book	/child::Books/child::Book[1]	null	1072
Name	/child::Books/child::Book[1]/child::Name	Sherlock Holmes-I	1073
Author	/child::Books/child::Book[1]/child::Author	Arthur Conan Doyle	1074
Book	/child::Books/child::Book[2]	null	1075
Name	/child::Books/child::Book[2]/child::Name	Sherlock Holmes-II	1076
Author	/child::Books/child::Book[2]/child::Author	Arthur Conan Doyle	1077
Book	/child::Books/child::Book[3]	null	1078
Name	/child::Books/child::Book[3]/child::Name	Sherlock Holmes-III	1079
Author	/child::Books/child::Book[3]/child::Author	Arthur Conan Doyle	1080

**FIG. 10C**

Node	XPath	Value	XPathAttributes
Books	/child::Books	null	1081
Book	/child::Books/child::Book[1]	null	1082
isbn="123"	/child::Books/child::Book[1]/attribute::isbn	123	1083
Name	/child::Books/child::Book[1]/child::Name	Sherlock Holmes-I	1084
Author	/child::Books/child::Book[1]/child::Author	Arthur Conan Doyle	1085
Book	/child::Books/child::Book[2]	null	1086
isbn="123"	/child::Books/child::Book[1]/attribute::isbn	123	1087
Name	/child::Books/child::Book[2]/child::Name	Sherlock Holmes-II	1088
Author	/child::Books/child::Book[2]/child::Author	Arthur Conan Doyle	1089
Book	/child::Books/child::Book[3]	null	1090
isbn="123"	/child::Books/child::Book[1]/attribute::isbn	123	1093
Name	/child::Books/child::Book[3]/child::Name	Sherlock Holmes-III	1095
Author	/child::Books/child::Book[3]/child::Author	Arthur Conan Doyle	1099

**FIG. 10D**



**FIG. 11**